Real-Time Embedded System Support for the BTeV Level 1 Muon Trigger

Michael J. Haney and the RTES Collaboration

Abstract – **The Level 1 Muon Trigger subsystem for BTeV will be implemented using the same architectural building blocks as the BTeV Level 1 Pixel Trigger: pipelined field programmable gate arrays feeding a farm of dedicated processing elements. The muon trigger algorithm identifies candidate tracks, and is sensitive to the muon charge (sign); candidate dimuon events are identified by complementary charge track-pairs. To insure that the trigger is operating effectively, the trigger development team is actively collaborating in an independent multi-university research program for reliable, self-aware, fault adaptive behavior in real-time embedded systems (RTES). Key elements of the architecture, algorithm, performance, and engineered reliability are presented.**

I. Introduction

THE proposed BTeV [1] Level 1 Trigger consists of similar yet independent Pixel and Muon Triggers, which use pipelined programmable logic and embedded processors to arrive at trigger opinions for each 132 ns beam crossing. These opinions are weighed by the Global Level 1 Trigger to produce a Level 1 trigger decision at the beam-crossing rate, with a resource-bounded latency of a few milliseconds.

The complexity of the system, in both hardware and software, raise serious concerns with respect to reliability. With thousands of embedded processors operating concurrently, undetected faults can corrupt the trigger results in a manner that can be extremely difficult to diagnose after the fact. And a local fault can have a global impact if not addressed. An aggressive approach to detection and fault adaptation is required.

The architecture and algorithm of the Level 1 Muon Trigger is presented, followed by a discussion of the tools that are being developed to address these reliability concerns.

II. BTeV Level 1 Muon Trigger

The Level 1 Muon Trigger processes wire-chamber hit data from 3 stations of 4 layers each. The stations (groups of layers) are separated by iron toroids. The layers consist of octants of common-oriented tubes. All tubes are transverse to the beam line. The R layers (views) have tubes perpendicular to the radial centerline of the octant; the stereo layers (U, V) have tubes oriented at +/- 22.5 degrees with respect to the R layer. The fourth layer in each station (denoted "S") is a duplicate R layer. There are approximately $36*10^3$ wire chambers in the detector.

Hit data is provided per tube, per 132 ns crossing, over 96 fiber optic cables. The Preprocessors of the Muon Trigger performs adjacent-tube hit compression and radial sorting. The radial sorting (from outer-most to inner-most) is an important performance aspect. Subsequent processing is performed on a time/space-available basis. Processing from the outer-most to the inner-most preferentially biases the result in favor of detecting high(er) P_T tracks, and ignoring the "ring of fire" hits in the tubes closest to the beam line. The Preprocessors will be implemented with field programmable gate arrays (FPGAs); 48 Preprocessor boards are required.

The compressed/sorted data are provided to a farm of embedded processors for analysis. Work is assigned to processor-groups (farmlets) on a round-robin basis; work is assigned to individual processors on a next-available basis. The specific processor architecture is unimportant to the Muon Trigger algorithm. For the sake of expediency, current development work employs a Texas Instruments DSP (TMS 320C6x family). The actual processor used will be selected at time of construction in FY2005. However, the number of processors per farmlet does have a significant influence on the buffer memory requirements of the farmlet to support round-robin assignment without backpressure or overflow. Queuing

Manuscript received October 29, 2003. This work was supported in part by the National Science Foundation Information Technology Research Program (number #ACI-0121658)

Michael J. Haney is with the University of Illinois, Urbana, IL 61801 USA (telephone: 217-244-6425, e-mail: m-haney@uiuc.edu).

The RTES Collaboration consists of: D. Beauregard, R. Iyer, Z. Kalbarczyk, Q. Liu, and L. Wang (Design and Validation of Reliable Networked Systems Research Group of the Center for Reliable and High-Performance Computing (CRHC) of the Coordinated Science Laboratory (CSL), University of Illinois); M. Haney and M. Selen (High Energy Physics Group of the Department of Physics, University of Illinois); D. Mosse and O. Shigiltchoff (Fault Tolerant Real-Time Systems (FORTS) Group in the Department of Computer Science, Link-to-Learn educational program, and College in High School educational program, University of Pittsburgh); R. Chopade, L. Hovey, M. Jung, D. Messie, and J. Oh (Department of Electrical Engineering and Computer Science, Syracuse University); S. Stone (High Energy Physics Group of the Department of Physics, Syracuse University); T. Bapty, S. Neema, S. Nordstrom, S. Shetty, S. Vashishtha, and D. Yao (ISIS (Institute for Software Integrated Systems), Vanderbilt University); P. Sheldon and E. Vaandering (BTeV Group, part of the High Energy Physics Group in the Department of Physics and Astronomy, Vanderbilt University); J. Butler and E. Gottschalk, (BTeV Collaboration of the Particle Physics Division, Fermi National Accelerator Laboratory); J. Kowalkowski and M. Votava (Computing Division, Fermi National Accelerator Laboratory); J. Appel (Fermilab Education Office, Fermi National Accelerator Laboratory).

studies indicate that 6 processors per farmlet will require buffer capacity for 20 events. At 2 Kbytes/event, this is at most a modest resource requirement.

Each octant is processed independently. Further, each view (*i.e.* a set of three layers, one from each station, with the same orientation) is processed separately. Consider as an example the R view (layers) from stations 1, 2, and 3, denoted as R_1, R_2, and R_3 respectively. A hit-tuple (r_1, r_2, r_3) is declared to be a muon track if the χ^2 fit of the track points back to the interaction point. Fortunately, it is not necessary to implement this expensive computation for every hit-tuple. By examining Monte Carlo generated collections of tube hits (and noise), it has been determined that a viable alternative is to perform a coordinate rotation and measure the distance of (r_1', r_2', r_3') from the plane of tracks ($R_3 = 27.69 - 1.26*R_1 + 2.20*R_2$) which originate from the interaction point. Further, the sign of the distance (above or below the plane) is an indication of the track curvature, and hence of the muon charge (+/-).

The implementation of this algorithm operates from the list of hits from R_2. For each r_2, a directed search is made for candidate R_3 hits. It is not necessary to search the entire R_3 layer, since hits outside of a bounded region (determined by the value of r_2) will exceed the limits for an adequate fit. For each (r_2, r_3), a directed search of R_1 then provides the tuple (r_1, r_2, r_3). This process is repeated for each view (the second R view (S), and both stereo views (U, V)), and for each octant. Since the track finding process also estimates charge (sign) from track curvature, dimuon events are detected by finding a pair of tracks with complementary charge (curvature).

Using a 150 MHz DSP, the above processing can be performed in 33 μs (mean execution time), which at 132 ns beam-crossing time, requires 227 processors. 250 are specified in the design, with farmlet buffering to accommodate the fluctuations in processing time.

It is no coincidence that the Muon Trigger is strikingly similar to Pixel Trigger [2]. Both employ an FPGA-rich preprocessing component, followed by an embedded processor farm for subsequent analysis. The primary differences between the triggers are that the Pixel Trigger has two FPGA layers (Preprocessor, and Segment Tracker) compared to the Muon Trigger's one, and the overall scale of the triggers. The Pixel Trigger is essentially 10 times larger than the Muon Trigger in board count, due to the increased computational demands of the Pixel Trigger. However, as a purposeful cost-savings plan, the Muon Trigger is being developed to utilize, wherever possible, the same hardware elements as the Pixel Trigger. Hence the Muon Trigger embedded processor farm and interconnection switch will be identical to the Pixel Trigger, except for a smaller size and different executable code.

III. Real Time Embedded System Support

To address concerns about the hardware and software complexity of the BTeV Trigger, an independent research initiative to study Real-Time Embedded Systems (RTES) was undertaken. The RTES Collaboration is funded by the National Science Foundation Information Technology Research Program, and is a multi-university group of computer scientists, electrical engineers, and physicists [3].

The mission of RTES (taken from the introduction to the NSF proposal), is "to develop methodologies and tools for designing and implementing very large-scale real-time embedded computer systems that:

1. achieve ultra high computational performance through use of parallel hardware architectures
2. achieve and maintain functional integrity via distributed, hierarchical monitoring and control
3. are required to be highly available
4. are dynamically reconfigurable, maintainable, and evolvable."

The RTES project consists of 3 primary software development branches (and an educational outreach element which will not be discussed here): a means for modeling and developing code for a large system of embedded processors, a strong self-protection mechanism to insure that code is executing, and a small-footprint solution for instrumenting and controlling execution throughout the system. These are each discussed in the following sections.

A. Model Integrated Computing (MIC)

Developing code for a large, complex system of FPGAs and embedded processors is a difficult task by itself. Incorporating a diversity of fault detection and mitigation strategies, coordinating the code generation for the processors, and modeling the system both prior to deployment as well as in operation is a daunting task.

To address this, Model Integrated Computing [4, 5] has been adopted. This approach utilizes a graphical representation the physical architecture, logical architecture, and the behavior of the system. Distinct (but integrated) drawings define the connectivity between hardware, as well as the computational tasks to be performed; the assignment of process to processor is thus orthogonalized from the hardware and software specification. Further, finite-state machine drawings can be used to simplify the specification of behavior, above and beyond the development of point-code to implement tasks.

Modeling is also provided. The described system can be well studied prior to operation. And with the incorporation of messaging from the embedded processors, the operating behavior of the run-time system can be displayed on the same set of drawings used to represent the system.

This approach is well suited to the operational choice of the C6x DSP, as a detailed hardware/software/kernel solution for this processor has already been developed.

B. Adaptive Reconfigurable Mobile Objects for Reliability (ARMORs)

ARMORs provide a strong solution-in-depth for reliable execution on full-rank computational elements [6, 7]. It is well suited to Linux operation (for which it was developed), but has been ported to Windows as well.

ARMORs fall into 3 primary types. The execution ARMOR is the most common. It provides direct monitoring and support of a "protected" application, including checkpointing for the application, and self-checking for the ARMOR itself. If the application hangs or crashes, the ARMOR detects this failure, and resets or restarts the application as appropriate.

To insure that the execution ARMOR does not fail, a supervisory Fault Tolerant Manager (FTM) ARMOR monitors subordinate ARMORs, resetting or restarting them as necessary.

And to insure that the FTM ARMOR does not fail, a "heartbeat" ARMOR monitors the FTM ARMOR, resetting or restarting it as necessary.

ARMORs are hierarchical and communicate via sockets. As such they can exist on any number of networked processors. Hardware failures can be handled by the FTM ARMOR by restarting the relevant execution ARMOR(s) and protected application(s) on any functioning processor with the resources needed by the application. ARMORs are also modular and highly configurable. A micro-kernel provides the basic operational behavior; user-written "elements" can be added to implement specific detection and recovery mechanisms.

In the context of BTeV, the ARMORs will see primary use in the Supervisory and Monitor subsystems, the various high level Control subsystems, as well as in the Level 2/3 Linux farm. Each of the Muon Trigger, Pixel Trigger, and Global Level 1 Trigger have their own Supervisor and Monitor subsystem (PTSM, MTSM, GL1SM) which is responsible for configuring FPGAs, loading code into embedded processors, monitoring operation, and detecting failures. Each of these "xxSM" subsystems will be a hierarchical network of a top-level host, intermediate-level regional managers, and low-level local managers to provide control and monitoring. Each of these will have ARMORs to perform or protect the control and monitoring function.

At the highest control level of BTeV, Slow Control and Run Control will utilize ARMORs to insure that the control applications and displays are executing. This capability has already been demonstrated [8].

And while not the topic of this paper, the Level 2/3 trigger farm which will use 2000+ Linux computers is a perfect candidate for ARMORs, providing direct protection of the L2/3 trigger application. The homogeneity of the farm and the ability of the FTM ARMOR to restart applications on remote processors immediately translates hardware failure into little more than a graceful loss of capacity.

C. *Very Light-weight Agents (VLAs)*

Complementary to the strength and resource-needs of ARMORs, VLAs provide a universal, minimal footprint solution for monitoring and control support [9]. These are small, fast, low-impact building blocks that can be incorporated in the embedded processors themselves, as well as at each level throughout the xxSM control system, and in the L2/3 Linux farm. VLAs handle communications to and from higher-level entities, and well as monitor and control lower-level entities. Common messaging and API presentation make these software elements available at every level of the system.

Because of their size and speed, VLAs are crucial to detection and mitigation in the embedded processor farm. VLAs have passive and active components. The passive components handle communications traffic from the trigger application in the embedded processor to the higher-level control processes, as well as control traffic from the xxSM host down to the embedded processors. The passive components also collect statistics and provide summary information to the higher levels.

The active VLA components are scheduled during processor idle time, and search for potential faults by direct testing (memory tests, checksums, etc.).

Like the ARMORs, the VLAs are a balance of basis code that provides the primary communications and control infrastructure, complemented by user-written modules for trigger-specific detection and mitigation.

Upon detection of a fault, the VLA can variously mitigate the problem locally (*e.g.* by reloading and restarting the trigger application to handle a hung application), and/or report the fault up the control chain for a higher-level decision to be made. A key feature throughout the system is the separation of capability from policy. The VLAs have sweeping powers to reset, restart, reload, and redirect, but whether and which of these powers is invoked is entirely controlled by a system-wide "authority vector" which defines the scope of who can decide what. At the most primitive setting, all faults are reported to the (human) operator, and all mitigation is directed from the operator's console. At the most sophisticated setting, each VLA and ARMOR acts autonomously, applying its best-guess at a fault mitigation solution, while reporting the fault, and the solution, to a higher level. It can be expected that different VLAs and ARMORs may interfere with each other by applying contradictory mitigation strategies. Effective system operation will require a balance between these extremes in policy, which will be implemented as differing values of the authority vector.

IV. SUMMARY

The Level 1 Muon Trigger will require a large number of programmable logic devices and embedded processors, which must function reliably or fail gracefully in order to deliver physics results. To insure this, an aggressive fault detection and mitigation research project (RTES) is developing modeling and development tools, protection modules, and instrumentation and control features, which in principle could be applied to any large-scale, embedded system.

A prototype of the RTES solution is in development with planned presentation at the SuperComputing 2003 [10].

V. REFERENCES

[1] www-btev.fnal.gov

[2] G. I. Cancelo, E. E. Gottschalk, F. V. Pavilcek, M. Wang, and J. Y. Wu, "Data Flow Analysis of a Highly Parallel Processor for a Level 1 Pixel Trigger," *IEEE Trans. Nucl. Sci.*, submitted for publication in these proceedings.

[3] www-btev.fnal.gov/public/hep/detector/rtes/index.shtml

[4] www.isis.vanderbilt.edu/Research/mic.html

[5] T. Bapty, S. Neema, S. Nordstrom, S. Shetty, D. Vashishtha, J. Overdorf, and P. Sheldon, "Modeling and Generation Tools for Large-Scale, Real-Time Embedded Systems," in the Proceedings of the 10th IEEE International Conference on Engineering of Computer Based Systems, Huntsville, AL, April 2003.

[6] www.crhc.uiuc.edu/DEPEND/projects-ARMORs.htm

[7] K. Whisnant, Z. Kalbarczyk, and R. Iyer, "A Foundation for Adaptive Fault Tolerance in Software," in the Proceedings of the 10th IEEE International Conference on Engineering of Computer Based Systems, Huntsville, AL, April 2003.

[8] L. Picolli, "Evaluation of RTES components in a large scale DAQ," in the Proceedings of the 13th IEEE-NPSS Real Time Conference, Montréal, Canada, May 2003.

[9] S. Tamhankar, J. Oh, and D. Mosse, "Design of Very Lightweight Agents for Reactive Embedded Systems," S. Tamhankar, J. Oh, Syracuse University, and D. Mosse, University of Pittsburgh, in the Proceedings of the 10th IEEE International Conference on the Engineering of Computer Based Systems, Huntsville, AL, April 2003.

[10] www.sc-conference.org/sc2003